

PROGETTO INGEGNERIA DELLA CONOSCENZA

A.A. 2022/23

House PRICE PREDICTION- DOCUMENTAZIONE

04/09/2023

Bernardino Cristallo - Matricola: 676879

b.cristallo3@studenti.uniba.it

Lorenzo Levanto – Matricola: 735404

l.levanto@studenti.uniba.it

## INDICE DEI CONTENUTI

- 01. INTRODUZIONE
- 02. INTRODUZIONE ALLE SCELTE PROGETTUALI
- 03. SCELTE PROGETTUALI
- 04. GUIDA ALL' UTILIZZO

### 01. INTRODUZIONE

House Price Prediction è un progetto software che si pone come obiettivo la predizione dei prezzi delle case americane. Date in input le caratteristiche della casa, il programma fornisce una predizione del prezzo e una predizione della probabilità tramite forme di apprendimento **supervisionato e non supervisionato**, e che il prezzo sia presente in una fascia di prezzi trami apprendimento **probabilistico**.

### 02. INTRODUZIONE ALLE SCELTE PROGETTUALI

Sviluppato in Python 3.7

Librerie utilizzate:

- Pandas
- Numpy
- Sklearn
- Matplotlib
- Tqdm
- Tkinter

Si è utilizzato un dataset estrapolato dal sito kaggle.com, chiamato “data.csv”, il file contiene i campi ‘**date**’ (data di registrazione), ‘**price**’ (prezzo), ‘**bedrooms**’ (numero camere da letto), ‘**bathrooms**’ (numero bagni), ‘**sqft\_living**’ (piedi quadrati vivibili),

---

'**sqft\_lot**' (piedi quadrati totali), '**floors**' (piani), '**waterfront**' (facciata su mare), '**condition**' (condizioni), '**sqft\_above**' (piedi quadrati calpestabili), '**sqft\_basement**' (piedi quadrati seminterrato), '**yr\_built**' (anno di costruzione), '**yr\_renovated**' (anno di ristrutturazione), '**street**' (via), '**city**' (città), '**statezip**' (CAP), '**country**' (nazione).

Al dataset sono state uniti i 2 campi, '**bedrooms**' e '**bathrooms**' in un unico campo '**rooms**' (numero delle stanze), per diminuire il numero delle features e incrementare il peso delle stanze a livello di predizione.

Il software genera un file "house.csv" che contiene il dataset risultante dalle operazioni appena descritte. Le istruzioni che creano tale file sono commentate, in quanto non più utili una volta che si crea il file. Tornerebbero utili in caso di aggiornamenti dei dataset utilizzati.

Per avere dati più accurati abbiamo deciso di eliminare i duplicati e i campi con i valori nulli dal csv, inoltre abbiamo deciso di convertire i 'piedi quadri' in 'metri quadri' poiché il sistema metrico decimale in Italia è quello dei metri.

Per ottimizzare la predizione del prezzo abbiamo scalato i valori del dataframe tra 0 e 1 e prima dell'**apprendimento supervisionato**, viene utilizzato l'**apprendimento non supervisionato** con operazioni di clustering, inoltre, in aggiunta viene utilizzata l'**apprendimento probabilistico** per predire la probabilità che il prezzo di una determinata casa possa essere presente in una determinata fascia di prezzo.

I modelli (di tipo random forest regressor) utilizzano come training-set una porzione (80%) del dataset, l'altra porzione (20%) è usata come test-set. La scelta di non utilizzare tutta la base di conoscenza come training-set è atta ad evitare l'overfitting e a valutare i modelli appresi utilizzando dati (test-set) non utilizzati per l'apprendimento. Invece per l'apprendimento probabilistico viene utilizzato il modello SGD (Stochastic Gradient Descent) Classifier per la sua accuratezza e affidabilità migliore.

Successivamente vi è l'utilizzo del K-fold, per suddividere il training set in k fold e addestrare/valutare il modello su ogni fold, nei 2 modelli di Grid Search utilizzati appunto per vedere quali sono le variabili migliori da utilizzare per entrambi i modelli.

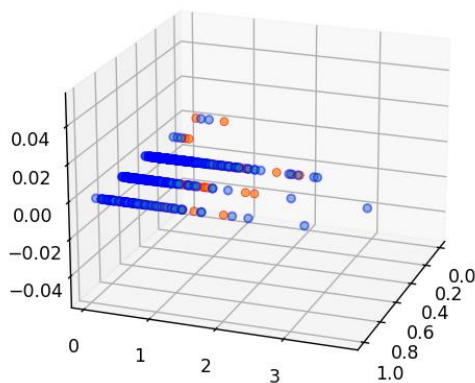
La valutazione della qualità del modello del Random Forrest si basa sul calcolo del punteggio medio **MAE** (Mean Absolute Error '110438.91') calcolato tra i prezzi predetti per le case del test-set e quelli reali, sul **MSE** (Mean Square Error '34133152382.54') calcolato sul test-set e sul calcolo del **r2-score** ('0.75' = 75%) in grado di calcolarci la percentuale di accuratezza delle predizioni. Inoltre, per la valutazione della qualità del modello probabilistico è basato sulla percentuale dell'**accuracy** del SGD Classifier (75,28%)

---

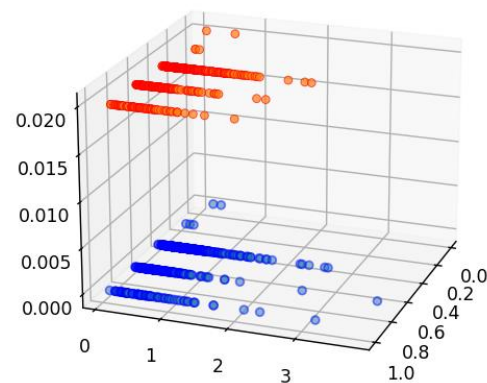
Le features selezionate sono i campi 'sqft\_living', 'sqft\_lot', 'floors', 'waterfront', 'view', 'condition', 'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated', 'street', 'city',

'country', 'rooms', i campi 'date' e 'statezip' si sono mostrati irrilevanti per la qualità della predizione di Price.

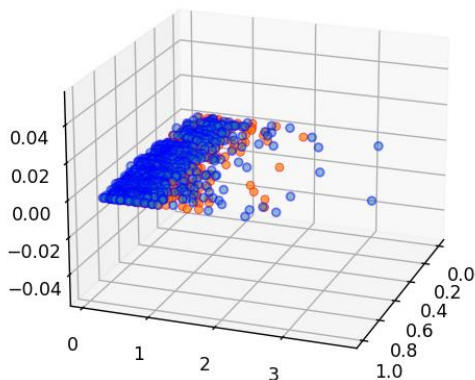
Per mostrare l'accuratezza della predizione per **ogni singolo feature** abbiamo inserito la stampa di grafici che dimostrano tale accuratezza (in blu i valori reali delle features e in rosso la loro predizione), le istruzioni che creano tale grafico sono commentate, in quanto non più utili una volta che si crea il grafico, di seguito qualche esempio rispettivamente delle seguenti features (sqft\_living, floors):



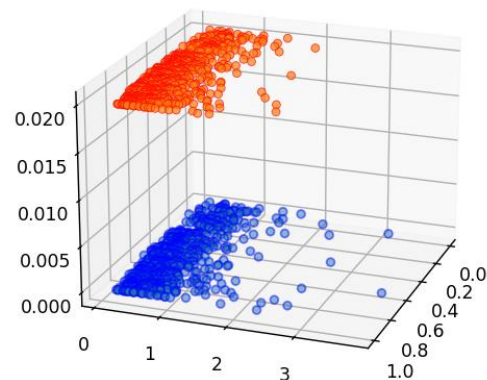
100



100



100



100

### 03. SCELTE PROGETTUALI

**Scelta di "IndexState = 2167" →** grazie l'utilizzo di un ciclo che partiva da un `indexstate = 0` e finiva a un `indexstate = 10.000` abbiamo cercato l'`indexstate` che ci restituisse il valore migliore a livello di predizione nel RandomForest.

**Scelta dei range da utilizzare nell'apprendimento probabilistico →** sono stati utilizzati le fasce di prezzo presenti nel codice a discrezione nostra dopo aver letto tutto il file `house.csv` e aver visto quali sono i prezzi più ricorrenti.

**Uso del `get_dummies` e del `LabelEncoder` →** Il `get_dummies` viene utilizzato sulla feature "city" per la creazione di nuove colonne dove ogni colonna rappresenterà ogni singola città e sarà valorizzata con 0 o 1 in base al valore effettivo di quella città (es. "city\_Seattle=1" | "tutte le altre città=0"); invece, `LabelEncoder` viene utilizzato sulle features "street, country" così da riuscire a trasformare le stringhe in valori labels (es. "street= 18810 Densmore Ave N" => "street = 42"), in modo tale che il modello operi solo su numeri piuttosto che su stringhe.

**Normalizzazione su tutte le features tranne "price" →** "price" non viene normalizzata poiché ci servirà il valore numerico intero per effettuare le varie predizioni, essa verrà discostata dal nostro modello essendo il nostro target.

**Apprendimento non supervisionato →** DBSCAN è un algoritmo di clustering che suddivide i dati in gruppi sulla base della loro densità, identificando punti densi come nuclei dei cluster e punti meno densi come rumore o outliers. Qui viene creato un oggetto "clusters" utilizzando l'algoritmo DBSCAN con i parametri `eps=0.9` (raggio della vicinanza) e `min_samples=3` (numero minimo di punti nel raggio per formare un cluster), i valori sono quelli di default. L'oggetto "clusters" viene addestrato sui dati di "prices\_x" e dopo l'addestramento con DBSCAN vi è la restituzione dell'etichetta di clustering assegnata a ciascuna istanza dei dati in "prices\_x". Successivamente viene aggiunta una nuova colonna chiamata "noise" al dataframe "prices\_x" e al dataframe "prices\_y" e vengono assegnate le etichette di clustering ottenute dall'algoritmo DBSCAN. Vengono selezionate solo le righe che hanno un valore di "noise" maggiore di -1. Questa operazione filtra le righe che non sono state assegnate a nessun cluster, quindi rimuove il rumore. E infine, La colonna "noise" viene rimossa dal dataframe "prices\_x" utilizzando il metodo "drop".

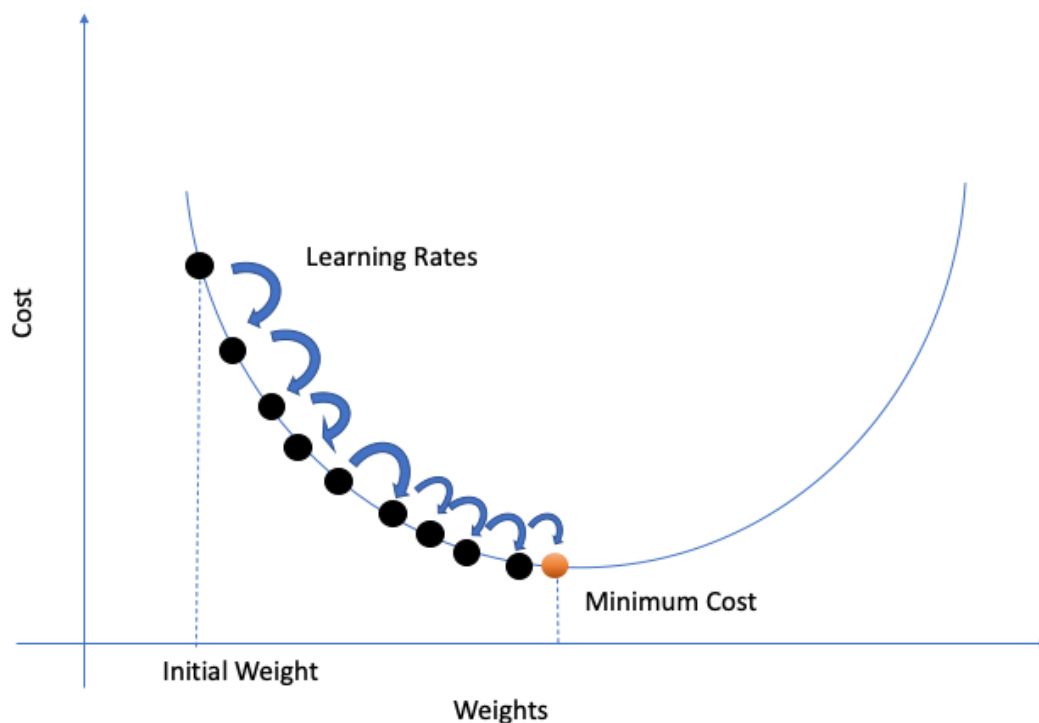
Riguardo al clustering, DBSCAN è un algoritmo di clustering "soft" perché assegna i punti a cluster o al rumore.

**Train/Test split →** la scelta di splittare il training set e il test set con la variabile "test\_size=0.2" è stata fatta dopo svariati tentativi e sapendo che è idealmente più corretto riservare 2/3 al training set e 1/3 al test set.

---

**SGD\_model** → La scelta del SGD Classifier è stata fatta dopo aver tentato di effettuare un apprendimento probabilistico tramite il Gaussian model e il Multinomial model ed aver avuto risultati poco attendibili poiché, il primo è un ottimo classificatore, ma non un buon stimatore di probabilità e invece il secondo ci dava semplicemente un Accuracny inferiore.

### Architettura:



Lo SGDClassifier si basa sull'algoritmo di ottimizzazione Stochastic Gradient Descent (SGD). Durante l'addestramento, esso regola iterativamente i parametri del modello (pesi e bias) usando mini-batch di dati di addestramento (Dataset) per minimizzare una funzione di perdita. Questo processo mira a trovare i parametri ottimali e a minimizzare il costo finale (Loss).

L'obiettivo dell'SGD è raggiungere il punto più basso, rappresentato da un pallino arancione nel grafico iniziando da un punto (peso) totalmente randomico (Initial Weight).

Successivamente, tramite un parametro denominato Learning Rate, questi pesi si spostano verso la ricerca del minimo globale. Questo parametro (rappresentato dalle frecce blu) è di vitale importanza, dato che se impostato ad un valore troppo alto, rischia di farci saltare completamente il punto più basso del costo, mentre se impostato ad un valore troppo basso, rischia di non farci mai raggiungere quel punto, muovendoci troppo lentamente.

L'SGDClassifier, di natura, applica come modello base o l'SVM (Support Vector Machine) o la Logistic Regression. Abbiamo utilizzato il secondo modello, dato che prevedeva l'applicazione di una funzione di decisione basata su probabilità.

### **Funzione di decisione:**

Il classificatore calcola una funzione di decisione per ogni istanza nel dataset (sample). Questa funzione è una combinazione lineare dei valori delle features e dei loro pesi corrispondenti (weights).

Matematicamente, per un problema di classificazione multiclass, può essere rappresentata come:  $w_1 * x_1 + w_2 * x_2 + \dots + w_n * x_n + b_i$

dove  $w_1, w_2, \dots, w_n$  sono i pesi,  $x_1, x_2, \dots, x_n$  sono i valori delle varie features e  $b_i$  è il bias di quella specifica classe.

### **Calcolo delle probabilità di classe:**

Le probabilità di classe vengono calcolate utilizzando la funzione di decisione, citata pocanzi. Queste probabilità vengono ottenute trasformando la funzione di decisione in un valore di probabilità usando la funzione di attivazione chiamata Softmax (data la classificazione multiclasse) che ci restituisce un valore compreso tra -1 e 1 che può essere interpretato come la probabilità del sample di appartenere a quella specifica classe.

La funzione softmax è definita come:

$$P(y=i | x) = \exp(\text{score}_i) / \sum(\exp(\text{score}_j) \text{ for } j \text{ in classes})$$

Dove:

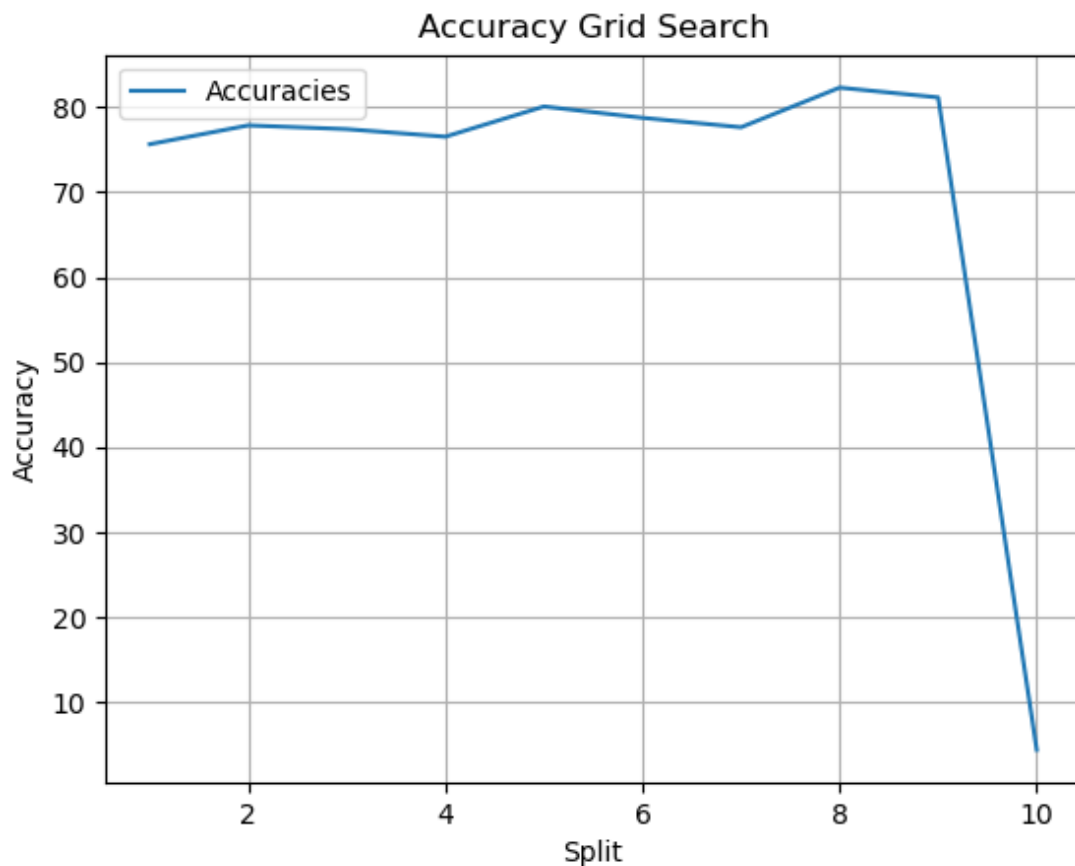
- **$P(y=i | x)$**  è la probabilità che l'istanza appartenga alla classe  $i$  dato l'input  $x$ .
- **$\exp(x)$**  è la funzione esponenziale.
- **$\text{score}_i$**  è il punteggio (funzione di decisione) per la classe  $i$ .
- **$\text{classes}$**  rappresenta l'insieme di tutte le classi.

### **Uso di predict\_proba()**

Quando si chiama il metodo **predict\_proba()** su un sample o un insieme di sample, il classificatore calcola le probabilità di classe in base ai loro valori delle feature e ai parametri del modello appresi (pesi). Il metodo restituisce un array in cui ogni riga corrisponde a un'istanza, e ogni colonna rappresenta le probabilità (tra -1 e 1) che quel sample appartenga a quella specifica classe. Nel nostro caso, avremmo 6 colonne, date le nostre 6 classi.

---

Inoltre, nel SGD è stato scelto il valore dello split n.8 “ $\alpha=0.0001$ ” dopo aver cercato tramite il grid search il miglior parametro da utilizzare. Di seguito il grafico indica il punto migliore che ci indica l'accuratezza migliore riguardante i valori da utilizzare e, in seguito, la stampa su terminale dei vari parametri con i vari split e accuratezze:



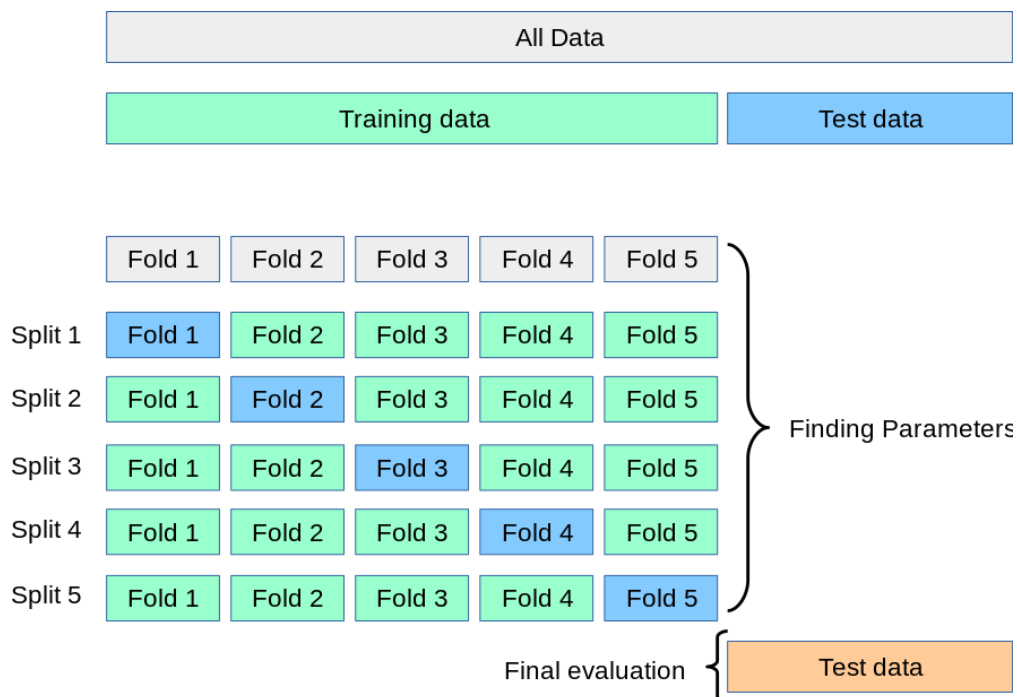


```

Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.1 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.1: 75.61%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.2 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.2: 77.83%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.3 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': False}
Accuracy split n.3: 77.38%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.4 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.4: 76.50%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.5 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.5: 80.04%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.6 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.6: 78.71%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.7 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.7: 77.61%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.8 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.8: 82.26%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.9 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.9: 81.11%
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Split n.10 | Best hyper-parameters per SGD: {'alpha': 0.0001, 'fit_intercept': True}
Accuracy split n.10: 4.44%

```

**Grid search** → nei grid search utilizzati sia per il SGD che per il RandomForest è stato scelto un KFold con 10 split poiché, nonostante di default sono impostati 5 split, abbiamo ritenuto più efficiente dividerlo in più Fold così da avere più precisione. Nello stesso algoritmo abbiamo impostato i valori sempre a nostra discrezione dopo aver letto le documentazioni della funzione. Di seguito un esempio di suddivisione in k-fold.



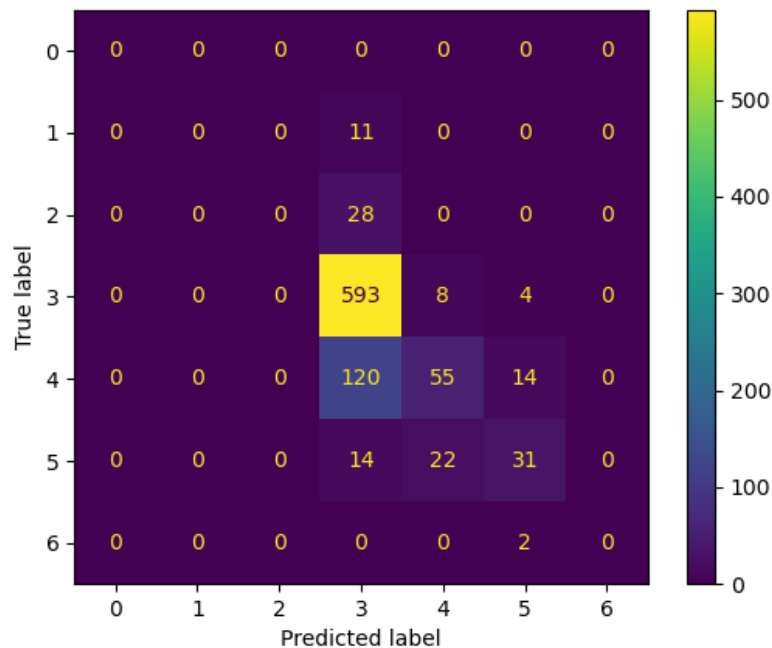
**Come funziona il grid search?** → Partendo dal training set si divide quest'ultimo in  $k$  parti (fold) e, analogamente, in  $k$  splits. Per ogni split viene utilizzato una parte di fold diversa come test set, in modo tale da trovare i migliori parametri per il modello. Infine, una volta trovati questi parametri, si procede alla classificazione con il test set originale. Per quanto riguarda la Grid Search sul SGD è stato impostato un valore di 'alpha': `np.arange(0.0001,0.0006,0.0001)` dove viene aumentato di 0.0001, per ogni split, il valore di alpha così da effettuare 100 fits; per quanto riguarda il fit intercept lo abbiamo impostato come una lista [True, False] in modo tale da provare il valore True e False per ogni valore di alpha. La stessa cosa è stata applicata per il Grid Search del RandomForest con valori diversi ma con la stessa metodologia.

**Accuracy del SGD** → Per verificare l'accuratezza del SGD lo abbiamo allenato sul training set e abbiamo lanciato l'algoritmo che ci restituisse le probabilità sul test set. Dato che i valori Groundtruth del test set corrispondevano ad un determinato range (es. "1,3,3,2,1,5,4"), abbiamo ricavato l'indice della probabilità maggiore (`np.argmax`) sulle predizioni di cui abbiamo parlato all'inizio. Infine, tramite un semplice confronto abbiamo controllato se gli indici predetti corrispondessero ai valori Groundtruth e popolato, di conseguenza, la variabile "correct". Per calcolarci l'accuracy finale del modello abbiamo diviso correct per il numero totale di sample presenti nel test set (`((correct/prices_y_test.size)*100)`).

Inoltre, abbiamo inserito una stampa del Classification Report con i valori di Precision(`tp/(tp+fp)`), Recall(`tp/(tp+fn)`), F1-score (`2 * (precision * recall) / (precision + recall)`), Support (sample per ogni classe) calcolati per ogni classe.

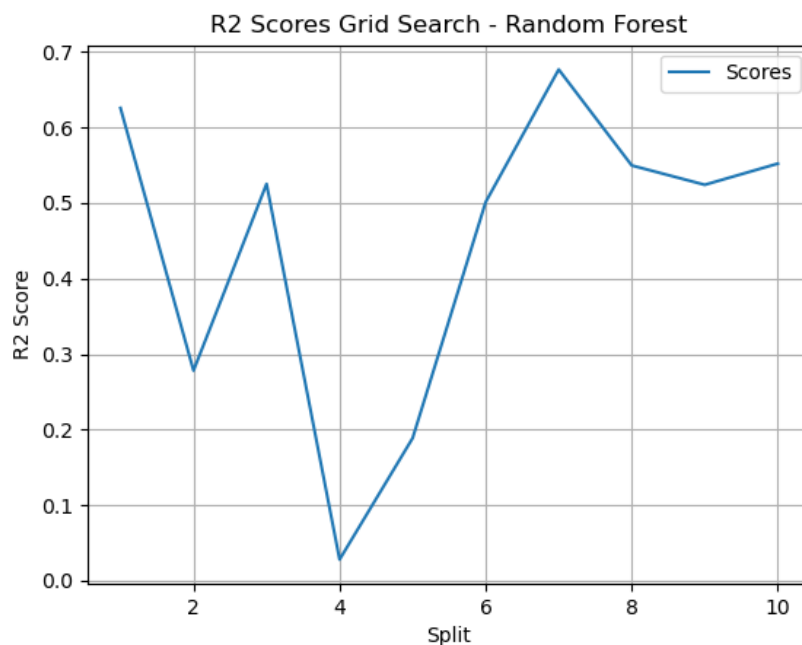
Classification report for SGD:				
	precision	recall	f1-score	support
0	0.000	0.000	0.000	0
1	0.000	0.000	0.000	11
2	0.000	0.000	0.000	28
3	0.774	0.980	0.865	605
4	0.647	0.291	0.401	189
5	0.608	0.463	0.525	67
6	0.000	0.000	0.000	2

Come tesi a supporto della Classification report abbiamo inserito la stampa del Confusion matrix che ci mostra quanti sample sono stati catalogati in quale classe, ad es. si può notare come 11 sample della classe 1 vengono classificati nella classe 3, oppure 14 sample della classe 4 vengono classificati come classe 5.



**RandomForest** ➔ Dopo aver provato vari modelli come il KNN o il LinearRegressor abbiamo accurato che il ForestModel era quello che più facesse al caso nostro, e dopo aver applicato anche su di esso il grid search abbiamo trovato i migliori parametri da utilizzare per la sua implementazione.

Come mostrato dal grafico seguente i migliori parametri coincidevano con l'R2-score corrispondente con lo split n.7:



## 04. GUIDA ALL' UTILIZZO

Aprire i file 'model.py' e 'house\_prediction.py', presenti nella cartella del progetto caricata sul repository, con un qualsiasi ambiente di programmazione (ad es. Visual Studio) che supporti Python (dalla versione 3.7).  
Eseguire house\_prediction, si visualizzerà tale interfaccia:

House Prediction in USA

### Selezione i valori

Nazione:	USA
Citta':	Algona
Via:	1 View Ln NE
Metri quadri Vivibili:	34.37383872166481
Metri quadri Lotto:	59.271646228167974
Metri quadri Seminterrato:	0.0
Metri quadri Calpestabili:	34.37383872166481
Anno di costruzione:	2014
Anno di restauro:	2014
Piani:	1.0
Affaccio sul mare:	0
Stanze:	2.0
Vista:	0
Condizione:	0
Scegli il modello:	Random Forest

Avvia Predizione


Selezionare i valori dei campi e premere 'avvia predizione' per visualizzare il prezzo predetto.

I campi dei metri quadri (ex. 'metri quadri vivibili', 'metri quadri lotto'...) sono gli unici campi che possono essere avvalorati senza dover per forza selezionare uno tra i valori presenti. I campi restanti vanno avvalorati scegliendo tra i valori disponibili, in modo da evitare che l'utente inserisca una nazione, regione, via ecc. inesistenti nel mondo reale. Nel caso in cui si volesse cambiare il metodo di predizione vi è

l'apposita combobox che permetterà la scelta avendo così gli output dedicati a quel tipo di apprendimento.

Di seguito qualche screen di qualche predizione fatta usando i valori di alcune case reali presenti nel csv:

Valore casa su house.csv = 235000€

 House Prediction in USA


## Seleziona i valori

Nazione:	USA
Citta':	Seattle
Via:	7542 21st Ave SW
Metri quadri Vivibili:	112.41174284652546
Metri quadri Lotto:	873.2813080639169
Metri quadri Seminterrato:	0.0
Metri quadri Calpestabili:	112.41174284652546
Anno di costruzione:	1949
Anno di restauro:	0
Piani:	1.0
Affaccio sul mare:	0
Stanze:	4.00
Vista:	0
Condizione:	2.0
Scegli il modello:	Random Forest

Avvia Predizione

**Il prezzo predetto è: 273890.53**

Valore casa su csv = 335000€

 House Prediction in USA


## Seleziona i valori

Nazione:	USA	▼
Citta':	Redmond	▼
Via:	2616 174th Ave NE	▼
Metri quadri Vivibili:	125.4180602006689	▼
Metri quadri Lotto:	237.82980304719436	▼
Metri quadri Seminterrato:	0.0	▼
Metri quadri Calpestabili:	125.4180602006689	▼
Anno di costruzione:	1976	▼
Anno di restauro:	0	▼
Piani:	1.0	▲▼
Affaccio sul mare:	0	▲▼
Stanze:	4.00	▲▼
Vista:	0	▲▼
Condizione:	3.0	▲▼
Scegli il modello:	Random Forest	▼

Avvia Predizione

**Il prezzo predetto è: 379132.26**

Esempio apprendimento probabilistico Valore casa su csv = 550000€

 House Prediction in USA

## Seleziona i valori

Nazione:	<input type="text" value="USA"/>
Citta':	<input type="text" value="Redmond"/>
Via:	<input type="text" value="9105 170th Ave NE"/>
Metri quadri Vivibili:	<input type="text" value="180.23039762170197"/>
Metri quadri Lotto:	<input type="text" value="975.4738015607581"/>
Metri quadri Seminterrato:	<input type="text" value="74.32181345224824"/>
Metri quadri Calpestabili:	<input type="text" value="105.90858416945375"/>
Anno di costruzione:	<input type="text" value="1976"/>
Anno di restauro:	<input type="text" value="1992"/>
Piani:	<input type="text" value="1.0"/>
Affaccio sul mare:	<input type="text" value="0"/>
Stanze:	<input type="text" value="6.50"/>
Vista:	<input type="text" value="0"/>
Condizione:	<input type="text" value="4.0"/>
Scegli il modello:	<input type="text" value="SGD"/>

Avvia Predizione

**Questo sample ha probabilità 78.56%  
di rientrare nella fascia (200000, 650000).**